06 | 执行：打造品质，要从头开始“闭环”

2019-11-09 雷蓓蓓

项目管理实战20讲 进入课程 >

讲述：雷蓓蓓

时长 13:35　大小 12.45M

你好，我是雷蓓蓓。今天我要跟你分享的主题是：打造品质，要从头开始“闭环”。

不知道你是否经历过这样的场景：你的团队历经一个多月没日没夜的奋斗，终于在圣诞节采购季来临前，完成了“黑五购物节”活动的所有功能，全部测试完毕，马上就要上线了。结果，产品负责人试用之后，皱着眉头说：“这……不是我想要的！” 你说，还有比这更悲惨的吗？

实际上，这种现象并不少见，有些产品经理还美其名曰“允许试错”。可是，最后做完了才发现不是自己想要的，早干吗去了呢？！

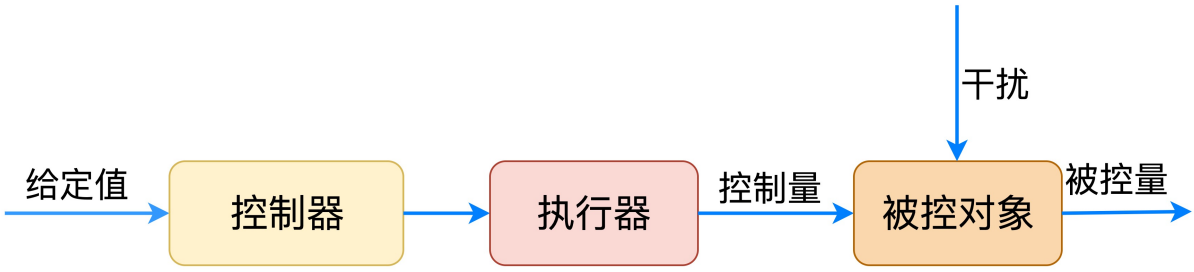
之所以会出现这种情况，有很多潜在的原因。比如，在互联网环境下，要弄清楚开发什么产品，准确把握并实现用户需求，对产品人员的要求其实非常高。对于互联网产品而言，从最

初的一个想法，到确定规模化的增长模式，通常要经历很多轮的螺旋式迭代，不断调整。

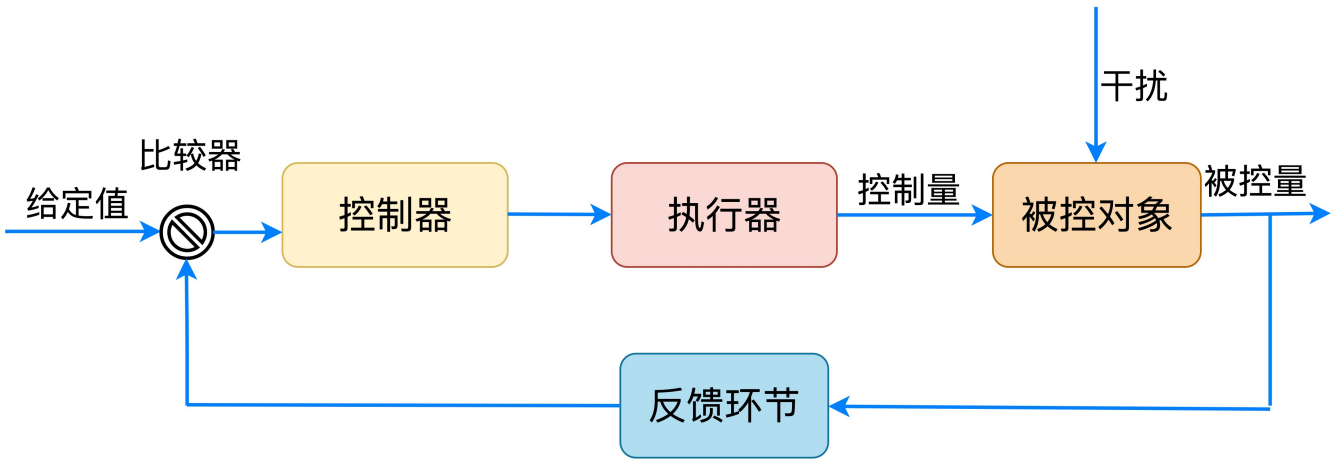
除了这个原因之外，更为常见的情况是，需求和设计根本没有经过严格的把关，就匆忙投入开发；同时，在传统的研发中间过程中，很难看到串连起来的功能效果，产品经理必须等到快上线了，才能看到和真实体验到可完整运行的产品。但是这个时候，再想大幅度修改产品，代价已经非常高了。

很多时候，我们确实没有办法一步到位，但合理试错，减少不必要的浪费，仍然是可以做到的。在项目执行的过程中，想要降低偏差、减少返工，**你就需要构建系统能力，在产品研发的整个过程中，建立起真正闭环反馈的产品验证机制**。

说到这里，我想先提一下，到底什么才是真正的闭环。我是学控制工程出身的，大学教科书里的一张展示开环和闭环系统的经典图片让我印象非常深刻。看了这张图，你就会明白，其实，**开环与闭环之间的关键差异，就在于有没有反馈环节，以及系统是否会根据反馈自动调节**。



开环控制系统



闭环控制系统

那么，既然闭环如此重要，在产品研发的整个过程中，到底有哪些实用的闭环验证方法呢？我来给你介绍三种最实用的方法。

方案评审（OARP 决策机制）

其实，需求评审、交互评审、视觉评审都是非常基本的闭环验证方法。我在留言区了解到，有些项目是从来不做需求评审和设计评审的，产品人员找某位开发单方面讨论一下，需求就算定下来了，可以直接往下走了，这就是典型的开环系统。

还记得我在刚开始举的“黑五购物节”的例子吗？那次返工，不仅让团队一个多月的辛苦打了水漂，还错过了最黄金的购物节时间。不得不说，这样的开环系统，在上线后出现偏差是很正常的，因为在这个过程中，根本没有任何矫正的机会。

要想执行中不走样，你就必须把方案评审做到位。而一说到评审，很多人会说，不就是组织一个会吗？大家就是坐在一起看一看，流于形式。No！你需要理解的是，**评审的精髓不在会，而在于背后的决策机制**。只有决策机制清晰，职责分工明确，方案评审才会有好的闭环效果。我来给你介绍一个典型的决策机制，叫作 OARP。

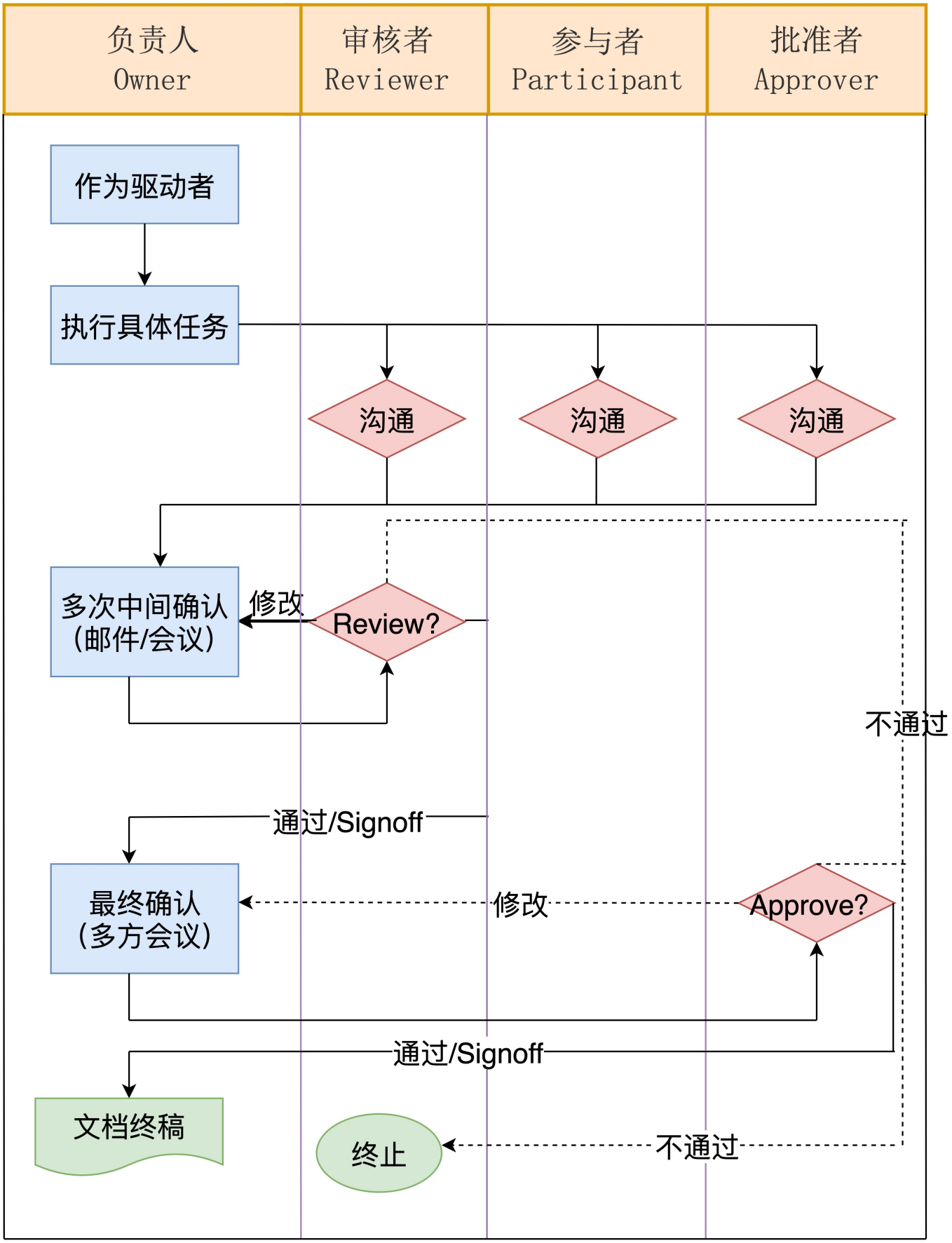
OARP 是 Owner、Approver、Reviewer、Participant 的缩写，分别对应四个关键角色：

负责人（Owner）：负责给出方案，组织各方讨论并推进做出最终的决定；

批准者 （Approver)：最终批准者；

审核者（Reviewer）：负责人和批准者挑选出的审核人。审核者有责任对文档进行讨论分析，并提出反馈意见，负责人必须重视并给予回复；

参与者 （Participant）：其他提供意见的人。参与者会收到文档的相关信息，可以对相关问题做出反馈。



这张流程图清晰地展示了一个典型的方案评审流程。OARP 是一套决策机制，你需要为项目中每一类方案的评审，确定明确的角色和职责，让各角色应享有的权利、应履行的职责，都得到规则上的保障，这样才能更好地确保方案质量，把后期的返工降到最低。

我把项目中常见文档所对应的 OARP 角色汇总在了下面这张表格里，你可以参考一下。

类型	Owner	Approver	Reviewer	Participant	确认方式
需求稿	策划人员	产品负责人	产品/设计/开发/测试Leader	项目组	会议+邮件
需求变更	策划人员	产品负责人	产品/设计/开发/测试Leader	项目组	任务流程
设计稿	设计人员	产品负责人	产品/设计/开发/测试Leader	项目组	会议+邮件
开发设计文档	开发人员	技术负责人	开发/测试/运维Leader	项目组	邮件
测试计划/测试用例	测试人员	技术负责人	开发/测试leader	项目组	会议

Bug Bash（Bug 大扫除）

Bug Bash，也叫 Bug 大扫除，最初来自微软，是指在项目开发里程碑的末期（比如 Beta 版发布前），划出一个专门的时间段，在这期间，所有参与项目的人员，集中全部精力一起来给项目找 Bug，目的是从各个维度衡量和体验产品。

这是一个非常有意思的活动，在网易也很受欢迎，在反复的实战应用中，我们对 Bug Bash 做了很多的改良，产生了很多有趣的变种。除了应用于测试阶段，我们发现，Bug Bash 还是进行产品闭环验证的一个非常有效的方法。通常，在版本的需求稿和交互稿完成之后，我会专门安排一段时间，组织团队成员一起，集中精力给需求稿和设计稿找问题。

记得我第一次组织这样的活动时，程序员们开心坏了！因为，他们终于可以让策划和设计们，也尝尝修 Bug 的滋味了！曾经，在一次活动的需求设计 Bug Bash 上，运营同学发现了现有产品的逻辑错误，避免了上线后优惠券发放的漏洞，为我们提前规避了很大的损失。通过这些 Bug Bash 活动，我们把产品验证环节大大地提前了，不仅达到了更好的体验效果，还有效地保障了上线质量。

那么，想要组织一场 Bug Bash 活动，该从哪些方面入手呢？

1. 时间：测试类的 Bug Bash，你可以选择在全面功能测试结束后，Bug 趋于收敛的时间段开展；需求设计类的 Bug Bash，一般会放在需求稿或设计稿完成后举行。这个活动需要一到两个小时的时间，你一定要提前排除所有干扰；

2. 地点：你需要设立一个单独的作战室，鼓励参与者自带笔记本，让他们尽可能集中精力做这一件事。同时，作战室可以提供一些零食和饮料，让活动更有氛围；
3. 参与者：除了研发和测试，产品、设计、市场、运营、销售等项目组不同角色的人群，都需要参与到这场活动中，这样你就可以获得更加丰富多维的视角；
4. 现场安排：你可以把现场反馈的问题直接贴在白板上，或者通过电子白板，来滚动更新 Bug 或建议的排名情况。需要注意的是，营造互动的氛围非常关键，如果因为空间受限，参与者做不到在同一个地点进行，你至少也要在通信群中实时更新排名情况的变化。
5. 活动结束：在活动结束后，要直接公示 Bug 或建议数，现场给奖品，并发邮件通报全组。最后，在对 Bug 进行去重后，还要分类定级，确定哪些 Bug 是必须修的，哪些 Bug 是改了会更好的。另外，千万不要忘记公示结果，明确修复计划。

关于活动的组织方式，你可以加入更多游戏化的玩法，这些都是为了最大程度地调动团队成员的参与热情，让问题在早期得到更好地暴露。你可以在一些重要版本中尝试这样的方法，与很多低效冗长的评审会相比，这个活动在避免返工方面，会有非常好的实战效果。

实际上，有时候，视觉稿我们也会拿出来这么做。曾经，某个已经运营了三四年的重量级产品，要在一个月之内完成全网的视觉改版，工作量巨大到难以想象，寻常做法很容易出问题，影响最终的用户体验。但时间非常紧张，我们根本来不及全部定稿，就必须开始并行开发，怎么办呢？

当时我就想到了 Bug Bash，不过这回不是做一次，而是每天都做！我们把计划拆分到按天交付的颗粒度，每天晚饭后的半个小时，大家会聚在一起，给刚刚做好的视觉稿找 Bug。开发和测试人员的早期参与，让很多遗漏的视觉问题在早期就得以发现，避免了后期的很多返工。后来，这个视觉改版非常顺利地上线了！

让我没想到的是，在项目组的回顾会上，每天晚上的 Bug Bash 活动，竟然高票当选最受欢迎的团队活动。仔细想想，Bug Bash 也是非常好的团建活动，我至今都很怀念那段虽然无比辛苦，但有吃有喝、充满欢笑的“革命岁月”。

冒烟用例评审

有同学留言问我说，当程序员说完成了某个模块的开发工作时，项目管理人员怎么知道 100% 完成了呢？其实，项目经理最怕听到的一个词，就是“**差不多**”。比如，差不多写完

了，差不多测好了，差不多可以发布了……每项工作都是差不多，可是一到测试环节，就会发现，其实还差得很远呢。

在上一讲中，我提到，做计划时要明确定义完成标准，而冒烟测试用例，可以说是开发和测试之间最基础的合约。

虽然“冒烟测试”这个概念很普遍，但是很多团队只是把它作为提测后的测试用例组，并没有真正发挥其合约的作用。要想避免掉进“差不多”的陷阱，在进入开发环节后，你需要安排专门的测试用例评审，让开发和测试对标准的冒烟用例集达成约定，这个约定就会成为进入测试的准入标准。

开发发起提测以后，测试人员就会依照这个标准用例集进行冒烟，并记录冒烟测试通过率，如果通过率不达标，就打回修正并再次提测。

如果你是在完全没有基础的团队中推行这个做法，可以**先从测试人员记录冒烟测试通过率开始**。接着，你要收集相关数据，和你的团队在回顾会上一起看看冒烟用例失败所导致的后期返工成本，讨论出一种更好的确保质量的做法。在我直接管理的项目中，冒烟通过率的要求通常是 100%，你可以选择从一个合适的起点开始，比如 80%，然后再一步步地逼近更好的提测质量。

总结

总结一下，今天，我给你介绍了在项目执行过程中，有效降低偏差、避免返工的三种闭环验证方法，包括方案评审、Bug Bash（Bug 大扫除），以及冒烟测试用例评审。

其实说到底，这些方法都是为了保证执行过程不走样。需要注意的是，你并不需要在每一个版本中把这些方法全都用上。你可以结合自己的项目情况，有步骤地开展优化，在核心的输出环节，设置合适的断点和关口，这样就能更好地把控全局了！

畅所欲言

这一讲中，我跟你分享的，都是我们自己在实战中积累下来的闭环验证方法。只有到了执行时，你才会发现，理想有多丰满，现实就有多骨感。我想请你来聊一聊，你在项目执行过程中踩过的最痛的“坑”。如果你可以再分享一些你的“爬坑指南”，那就更好了。

欢迎你畅所欲言，我在留言区等你，也欢迎你把文章分享给你的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　05 | 规划：排除计划中的“延期地雷”

精选留言 (15)

写留言

enjoylearning
2019-11-09

我们的设计稿都是产品一个人敲定，设计人员直接受产品经理管理。现在开需求会甚至都没有原型只有需求描述就开搞了，而且口头禅是 这个需求很简单，研发叫苦不迭

2

未未的未来
2019-11-09

坑很多，例如按照研发流程经过了立项评审，包括对需求和设计可行性的评审，领导也有参与，最后项目上线，没有达到客户预期。并不反对流程，但是在执行流程的过程中有些流于形式了，并未引起所有人员的重视，感觉公司有要求，就走走过场。
最后，除了问题，都得挨批，研发还得返工，这中间的帐就是没算没明白，总认为流程耽误时间，回头一看，欠的帐总是要还的，只是时间未到而已。

展开

1

maks

2019-11-09

我作为一个程序员，有一次接到需求之后。
按照流程需求分析，然后编写需求实施方案经邮件通知。
结果在开发完成，技术测试后。到业务测试环节，业务人员却说： “这不是我要的效果”
。
最可笑的是我们在核对需求都时候，却发现我从项目经理那拿到的是一个旧版本的需求...

展开

作者回复: 反向验证，从我做起，嗯!

1

Cy23

2019-11-11

尤其是改外包项目的时候，坑多的无法想象

展开

海罗沃德

2019-11-11

在敏捷開發團隊如何落實這些閉環？每個sprint時間都很短，沒有多餘時間進行這樣的閉環測試

有心猴

2019-11-11

最坑的就是工期倒推,产品的东西你可以评审,评审不合格就改,可是占用的都是开发的时间,最终结果是产品文档改的合格了,工期到了,是开发没有完成.

展开

穷查理

2019-11-10

周末下午，阳光明媚，怀孕五个多月的老婆在旁边床上睡觉，睡的很安详。看了一眼老婆熟睡的胖胖的脸，能给她带来这份安详，我感觉很欣慰。

盯着电脑思考了这个问题一个多小时。想了写，写了又删，删了又想，想到了项目中的很多坑，以及迈过一个个坑痛苦的过程，同时也想到了大领导跟我说的一句话（他说：因为我们不专业），...

展开

作者回复: 被你的文字感动到.....你一定会成为专业的项目经理，因为你遇到的每个坑，都带着一份礼物，哈哈，祝福你！

Hunter Liu

2019-11-10

任何功能必须让产品在各个环节多参与，多看效果，多与开发沟通，多让设计看Demo，因为每个人分工不同，思维方式不一样，产品人员是产品思维，设计人员注重细节和整体美感，开发人员是工程思维，当你思维方式不同，做出的东西可能就会有偏差，如果都弄好再看，是很难达成各方都满意的效果，为了避免后续大量返工，各种怨言，还是前期多参与多互动的好，不然，以我的经验，日程肯定是延期了，各方还都觉得输出了无用功...

展开

作者回复: 对的，早期多方视角的参与最重要

程序员人生

2019-11-09

网上曾有过这么个段子，产品经理提给开发一个需求，app的皮肤要根据用户手机壳的颜色改变而改变，不知道这个公司有没有需求评审这个环节

展开

程序员人生

2019-11-09

很多公司的前期评审更多地只是一个形式，并没有起到很好的效果。所以到后期问题都暴露出来，然后大家都在加班.....

展开

like_jun

2019-11-09

bug bash还没尝试过。

展开

威

2019-11-09

能否解析一下冒烟测试，以及和end2end测试有什么区别

杨家二少爷

2019-11-09

最大的坑就是甲方不同模块的业务人员沟通不充分，A模块上线不评估此需求功能其他模块是否有影响，导致上线后此功能一堆问题

展开

杨家二少爷

2019-11-09

期待蓓蓓姐做一场直播

展开

> 作者回复: 蓓蓓姐正在超级马力通宵备课状态……
>
> 现在，做好课程是第一位。直播会有的，让子弹飞一会～

javaadu

2019-11-09

对于超过两周的项目来说，PRD评审、系分评审、测分评审、上线评审，看起来是增加了流程复杂度和工作量，但是实际上这些事情和时间本来就是要花的

展开